Circuit Breaker - 2022

**Circuit breaker** is a pattern used to detect the failure and prevents an application from repeated/cascading failures.

Different States of Circuit Breaker. The purpose of the Circuit Breaker pattern is different than the Retry pattern. The Retry pattern enables an application to retry an operation in the expectation that it'll succeed. A circuit breaker acts as a proxy for operations that might fail.

* Closed
* Open
* Half Open

**Closed State**

When everything is normal, the circuit breaker remains in the closed state and all calls pass through to the services. **When the number of failures exceeds a predetermined threshold the breaker trips**, **and it goes into the Open state**.

**Open** – The request from the application fails immediately and an exception is returned to the application.

**Half-Open** – After a timeout period, the circuit switches to a half-open state to test if the underlying problem still exists. If these requests are successful, it's assumed that the fault that was previously causing the failure has been fixed and the circuit breaker switches to the Closed state. If any request fails, the circuit breaker assumes that the fault is still present so it reverts back to the Open state

**Circuit Breaker using Resilience4J**

**Main application**

@EnableFeignClients

@SpringBootApplication

**public** **class** SpringBootResilience4jApplication {

**public** **static** **void** main(String[] args) {

**SpringApplication.*run*(SpringBootResilience4jApplication.class, args);**

}

}

**Feign Client**

@FeignClient(name = "demo", url = "http://localhost:9090/")

**public** **interface** ServiceClient {

@GetMapping(path = "/information/info", name = "infoName")

String getDataInfo();

}

**Service**

**public** **interface** SomeService {

String getData();

}

**Relevant Portion in pom.xml**

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-openfeign</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-circuitbreaker-resilience4j</artifactId>

</dependency>

**Service Implementation**

@Service

**public** **class** SomeServiceImpl **implements** SomeService {

@Autowired

**private** ServiceClient client;

**@CircuitBreaker(name = "demoName", fallbackMethod = "getAlternateInfo")**

@Override

**public** String getData() {

**return** client.getDataInfo();

}

**public** String **getAlternateInfo(Throwable t)** {

**return** "Currently service is not available, try after sometime";

}

}

**Controller**

@RequestMapping("/external")

@RestController

**public** **class** SomeController {

@Autowired

**private** SomeService service;

@ResponseBody

@GetMapping("/info")

**public** String getExternalInfo() {

**return** service.getData();

}

}

**Circuit Breaker using Hystrix**

**Main application**

**@EnableCircuitBreaker // <--- Used for Hystrix**

@EnableFeignClients

@SpringBootApplication

**public** **class** SpringBootResilience4jApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringBootResilience4jApplication.**class**, args);

}

}

**Service Implementation**

@Service

**public** **class** SomeServiceImpl **implements** SomeService {

@Autowired

**private** ServiceClient client;

**@HystrixCommand(fallbackMethod = "getAlternateInfo")**

@Override

**public** String getData() {

**return** client.getDataInfo();

}

**public** String getAlternateInfo(Throwable t) {

**return** "Currently service is not available, try after sometime";

}

}

**Dependency in pom.xml**

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-hystrix</artifactId>

<version>2.2.10.RELEASE</version>

</dependency>